REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 2-25-97	3. REPORT TYPE AND DATES COVERED Final 7-16-94 to 11-15-96

4. TITLE AND SUBTITLE Program Partitioning and Scheduling on Hierarchial Systems	5. FUNDING NUMBERS DAAH04-94-G-0306
6. AUTHOR(S) Dharma P. Agrawal	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Electrical and Computer Engineering Box 7911, North Carolina State University Raleigh, NC 27695-7911 ATTN. Prof. Dharma P. Agrawal	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P. O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARO 32315.10-MA

11. SUPPLEMENTARY NOTES
The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Efficient utilization of high-performance computers require good parallelism detection and program partitioning techniques followed by efficient scheduling of partitioned tasks. In this work, we address issues in parallelism specification and detection, particularly related to Object-Oriented(OO) programs. We have proposed solutions to overcome inheritance anomaly in Concurrent OO Languages. We have also proposed a novel type-inference mechanism for static type determination of objects in OO programs and have developed a precise call-graph construction technique. Moreover, we have developed efficient task scheduling algorithms which produce an optimal schedule given sufficient number of processors. The duplication-based scheduling algorithm scales down nicely if number of available processors is not sufficient.

19970321 021

14. SUBJECT TERMS Parallelism detection, partitioning, scheduling, object-oriented.	15. NUMBER OF PAGES 5
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Program Partitioning and Scheduling on Hierarchical Systems

Dr. Dharma P. Agrawal
Department of Electrical & Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911.
Ph: (919)-515-3984
Fax: (919)-515-5523
E-Mail: dpa@ncsu.edu.

Abstract

Efficient utilization of high-performance computers require good parallelism detection and program partitioning techniques followed by efficient scheduling of partitioned tasks. In this work, we address issues in parallelism specification and detection, particularly related to Object-Oriented(OO) programs. We have proposed solutions to overcome inheritance anomaly in Concurrent OO Languages. We have also proposed a novel type-inference mechanism for static type determination of objects in OO programs and have developed a precise call-graph construction technique. Moreover, we have developed efficient task scheduling algorithms which produce an optimal schedule given sufficient number of processors. The duplication-based scheduling algorithm scales down nicely if number of available processors is not sufficient.

Foreword

A wide variety of high-performance computers with radically new architectures have been evolving rapidly over the last few years. This stupendous computing power is provided by machines ranging from networked workstations to superscalar or VLIW machines. Particularly noticeable among these architectures are the distributed memory machines(DMM) whose use can be seen in a wide variety of applications including fluid flow, weather modeling, database systems etc. These advancements in hardware have put demands on the software community to develop efficient concurrent software systems so that this enormous computing power could be utilized to the maximum possible extent. This requires development of suitable programming paradigms, parallelism detection techniques and efficient task partitioning and scheduling strategies. The task partitioning algorithm partitions the application into separate tasks by detecting the parallelism in the program and represents them in the form of a Directed Acyclic Graph (DAG). After the application is transformed to a DAG, the tasks are scheduled onto the processors.

On the other hand, on the language front, the use of Object-Oriented(OO) languages has become widespread in every field of computing ranging from application programming to operating systems. The OO paradigm provides the tools and facilities for developing softwares that are easier to build, extend, reuse, modify, and maintain.

This work deals with detecting parallelism and automatically partitioning the applications, especially keeping in view OO programs; and then scheduling the partitioned tasks efficiently to reduce overall execution time.

Statement of the Problem Studied

Performing a parallel computation on a DMM can be done in two steps. The first step deals with specifying parallelism and detecting parallel tasks in an application. The second step consists of scheduling these parallel tasks. We discuss some issues involved with these two steps in detail below.

1. Parallelism Specification and Detection: Properties like inheritance and dynamic binding of objects pose numerous hurdles in correctly specifying and detecting parallelism in OO programs. One approach to specify parallelism is to use a concurrent OO language with constructs for denoting concurrency. But this leads to a major inconsistency named inheritance anomaly forcing the programmer to redefine classes. A major goal of this work is to solve inheritance anomaly. Another hurdle in the path of detecting parallelism is the lack of program point specific type information. It has been proved that static type determination for C++ is NP-hard. With insufficient type information for static analysis, many of the traditional optimization and parallelizing techniques are rendered useless for OO languages. One prime objective of this work is to propose a solution to this type determination problem. In order to expose the parallelism in a program to the maximum extent, an automatic compiler must perform control and data flow analysis beyond procedural boundaries. The backbone of such an interprocedural analysis is a precise call-graph. But due to lack of exact type information, construction of a precise call-graph for OO programs is not possible. This work also intends to find out viable alternative methods for precise call-graph construction for OO programs.

2. Scheduling: Task scheduling is one of the key elements in any DMM. One of the major limitations of DMMs is the high cost for interprocessor communication which can be reduced by an efficient scheduling algorithm. A primary goal of this work is to reduce the overall execution time which includes communication as well as execution time. It has been proven that optimal scheduling of tasks onto DMMs is an NP-complete problem, and several heuristic based approaches have been explored. This work intends to investigate the trade-off between the schedule length and the required number of processors. A prime objective is to find the optimium schedule for DMMs if certain conditions are satisfied and if adequate number of processors are available. However, the system might not have the required number of processors to produce the optimal schedule. Therefore, it is necessary to develop an algorithm which scales down to produce the optimal algorithm for a given number of available processors.

Significant Accomplishments

1. We have mentioned in the statement of the problems the necessity of overcoming inheritance anomaly for concurrent OO languages. We have proposed a task-parallel language based on C++ called *CORE* which solves inheritance anomaly.

2. We have discussed earlier that it is extremely important to know the exact type of an object at a particular program point to apply any parallelization technique. The existing type inference algorithms fail to address this problem adequately enough for OO programs. We have proposed an approach named *SSAInfer* which transforms programs into static single assignment (SSA) form before any type inference mechanism is applied. This SSA-based approach combined with other constraint-based type inference methods, produces program-point specific as well as sharper types. Another property of *SSAInfer* is that it is language-independent and can be used in conjunction with other existing methods.

3. We have also proposed another type analysis approach named *ITA*. *ITA* performs an incremental type analysis within the constraint-based framework for restoring correct type information for type variables after program transformations. *ITA* blends very well with other compiler optimizations, such as constant propagation, in improving types.

4. Besides type analysis methods, we have also explored complementary approaches to do aggressive parallelization of OO programs. As mentioned earlier, failure to build a precise call-graph hinders interprocedural analysis of OO programs. Apart from lack of type information, this is also due to presence of virtual functions and inheritance. We have proposed and implemented an algorithm for precise call-graph construction using class hierarchy analysis. We have run this algorithm on a suit of benchmark programs which shows considerable improvement in the preciseness of the call-graph.

5. A code generation framework and run-time system has been developed for Sisal compiler which maintains the static and dynamic ownerships at every processor to avoid communication overhead on ownership information. The compiler has been targeted to Intel Touchstone i860 systems. The speed-ups in some cases are low compared to the required number of processors, because an inverted-tree-type parallelism is present in most Sisal programs. This type of parallelism is the cause of overall high processor demand with relatively low speed-ups.

6. We have proposed a Search and Duplication Based Algorithm(SDBS) with a complexity of $O(V^2)$, where V is the number of tasks. The input to the algorithm is the tasks represented in the form of a Directed Acyclic Graph (DAG) and it produces the optimal schedule given sufficient number of processors and if certain conditions are met.

7. We have also developed a Scalable and Task Duplication based Scheduling(STDS) algorithm which is an improvement upon SDBS in terms of scalability. STDS still has an worst case complexity of $O(V^2)$ where V is the number of nodes in the DAG. This algorithm initially generates clusters similar to linear clusters and uses them to generate a new schedule. It uses the concept of duplicating critical tasks to arrive at the optimal schedule. If the number of available processors is less than the number of linear clusters, the algorithm scales down nicely and still produces a near optimal schedule. The algorithm has been applied to some practical DAGs like Cholesky decomposition and its performance shows improvement over existing scheduling techniques. The numbers obtained show that with decreasing number of available processors, the schedule length goes up in discrete steps. This can be explained by the fact that, as number of processors go down, for a while the critical path remains unaffected but after a while, with more merging of lists, the critical path length goes up.

Refereed Publications

1. S.S. Pande, and D.P. Agrawal, "Run-Time Issues in Program Partitioning on Distributed Memory Systems," *Concurrency: Practice and Experience, Special Issue on Scheduling*, Vol. 7, No. 5, Aug. 1995, pp. 429-454.

2. S. Darbha and D.P. Agrawal, "SDBS: A Task Duplication Based Optimal Scheduling Algorithm," *In Proceedings of Scalable High Performance Computing Conference*, May 1994.

3. S. Darbha and D.P. Agrawal, "A Task Duplication Based Optimal Scheduling Algorithm for Variable Execution Time Tasks," *In Proceedings of International Conference on Parallel Processing*, August 1994.

4. S. Darbha and D.P. Agrawal, "A Fast and Scalable Scheduling Algorithm for Distributed Memory Systems," *Proc. 7th IEEE Symposium on Parallel and Distributed Processing*, San Antonio, TX, Oct. 25-28, 1995, pp. 60-63.

5. S. Darbha and D. P. Agrawal, "Optimal Scheduling Algorithms for Distributed Memory Machines", Submitted to *IEEE Transactions on Parallel and Distributed Systems.*

6. S. Darbha and D. P. Agrawal, "A Task Duplication based Scheduling Algorithm for Distributed Memory Systems", Submitted to *Journal of Parallel and Distributed Computing.*

7. S. Darbha, "Task Scheduling Algorithms for Distributed Memory Systems", *PhD. Thesis*, Department of Electrical and Computer Enginnering, North Carolina State University, Raleigh, NC, 1995.

8. S. Kumar and D.P. Agrawal, "A Class Based Framework for Reuse of Synchronization Code in Concurrent Object-Oriented Languages," *International Journal of Computers and Their Applications*, Vol. 1, No. 1, Aug. 1994, pp. 11-23.

9. S. Kumar, "Issues in Parallelizing Object Oriented Programs," *Proc. Intn'l Conf. on Parallel Processing Workshop on Challenges for Parallel Processing*, Oconomowoc, WI, Aug. 14, 1995, pp. 64-71.

10. S. Kumar, D.P. Agrawal, and S.P. Iyer, "An Improved Type-Inference Algorithm to Expose Parallelism in Object-Oriented Programs," *Proc. of the Third Workshop on Languages, Compilers, and Run-Time Systems for Scalable Computers*, Troy, New York, May 22-24, 1995, pp. 283-286.

11. S. Kumar and D.P. Agrawal, "ITA: an Incremental Type Analysis Approach for Object-Oriented Programs," *Technical Report*, Department of Electrical and Computer Enginnering, North Carolina State University, Raleigh, NC, 1996.

12. S. Kumar, "Specification and Detection of Parallelism in Object-Oriented Programs," *PhD. Thesis*, Department of Electrical and Computer Enginnering, North Carolina State University, Raleigh, NC, 1996.

13. D. Bairagi, S. Kumar and D.P. Agrawal, "Parallelizing OO Programs: Precise Call-Graph in the Presence of Virtual Functions," *Workshop on Interaction between Compilers and Computer Architectures*, San Antonio, TX, Feb. 1997.

Awards and Honors

A **Meritorious Service Award** was given to Dharma P. Agrawal by the IEEE Computer Society.

List of Participating Scientific Personnel Who Earned Advanced Degrees

1. Sekhar Darbha, PhD, 1995.
2. Sandeep Kumar, PhD, 1996.